17/1/2023

Question 1 .... *5 points*

Prove or disprove that:

(a) [1 point] $n(n+1)/2 \in O(n^2)$ correct

$$\frac{n(n+1)}{2} = \frac{n^2+n}{2} = \frac{n^2}{2} + \frac{n}{2} \leq n^2 \quad \forall\ n \geq 1$$

by using ployNomial thereome $\Leftarrow C = 1 \quad n_0 = 1$

(b) [1 point] $n(n+1)/2 \in \Omega(n^2)$ by using polynomial thereom $n_0 = \left|\frac{\sum_{i=0}^{k-1} a_i}{a_k - C}\right| \quad c = \frac{1}{4}$

$$= \frac{n^2}{2} + \frac{n}{2} \geq \frac{1}{4}n^2$$

$$\frac{n^2}{2} + \frac{n}{2} \geq \frac{1}{4}n^2 \quad \forall n \geq 2$$

$$n_0 = \frac{\frac{1}{2}}{\frac{1}{2} - \frac{1}{4}} = \frac{4}{2} = 2$$

$C = \frac{1}{4} \quad n_0 = 2$ correct

(c) [1 point] $n(n+1)/2 \in \Theta(n^2)$

frome previous Q a,b we can see that $C_1 = \frac{1}{4} \quad C_2 = 1 \quad \max(n_0[1,2]) = 2$

$n_0 = 2$

Correct

(d) [1 point] $n(n+1)/2 \in o(n^2)$

Uncorrect

by defintion of small-o ~~that~~ All constants $c > 0$ should accepet $o(n^2)$

$$\frac{n^2}{2} + \frac{n}{2} < n^2 \quad \text{when} \quad c = 1 \quad n = 1$$

$$1 \not< 1 \qquad \text{So } \frac{n(n+1)}{2} \notin o(n^2)$$

(e) [1 point] $n(n+1)/2 \in \omega(n)$

by definition small-ω All constants should accept the $\omega(n)$ and this is true

$\frac{n^2}{2} + \frac{n}{2} > n \quad \forall n \geq 1 \ \forall c > 0$

correct

Question 2 . . . . 4 points

For each of the following functions, indicate the class $\Theta(g(n))$ the function belongs to. Use the simplest $g(n)$ possible in your answers.

(a) [1 point] $f(n) = (n^2+1)^{10}$

$f(n) \in \Theta(n^{12})$ $\quad n^{20}$ $\quad \times$

-1

(b) [1 point] $f(n) = 2n\log(n+2)^2 + (n+2)^2\log(\frac{n}{2})$

$f(n) = 4n\log(n+2) + (n^2+4n+4)(\log n - \log 2)$

$f(n) = 4n\log(n+2)\ n^2\log n + 4n\log n + 4\log n$

$f(n) \in \Theta(n^2\log n)$

(c) [1 point] $f(n) = \sqrt{10n^2+7n+3}$

$f(n) \in \Theta(n)$

(d) [1 point] $f(n) = \log^2(n).\log(n^2)$

$f(n) \in \Theta(\log^3(n))$

Question 3 . . . . 2 points

Solve the following recurrence relation by using the backward substitution method.

$$\begin{cases} T(n) = 3T(n/5) + n & \text{for } n > 1 \\ T(1) = 0 \end{cases}$$

$T(n) = 3T(n/5) + n$

$T(n) = 3\left[3T\left(\frac{n}{5^2}\right) + \frac{n}{5}\right] + n$

$T(n) = 3^2\,T\left(\frac{n}{5^2}\right) + \frac{3}{5}n + n$

$= 3^2\left[3T\left(\frac{n}{5^3}\right) + \frac{n}{5^2}\right] + \frac{3}{5}n + n$

$= 3^3\,T\left(\frac{n}{5^3}\right) + \left(\frac{3}{5}\right)^2 n + \left(\frac{3}{5}\right)n + n$

$\vdots$

after k step $\quad T(n) = 3^k\,T\left(\frac{n}{5^k}\right) + \sum_{i=0}^{k-1}\left(\frac{3}{5}\right)^i n$

$\frac{n}{5^k} = 1 \qquad = 3^k\,T\left(\frac{n}{5^k}\right) + n\left(\frac{1-(\frac{3}{5})^k}{1-\frac{3}{5}}\right)$

$k = \log_5 n$

$T\left(\frac{n}{5^k}\right) = 0 \qquad = 3^k.\,T\left(\frac{n}{5^k}\right) + \frac{5}{2}n\left(1 - \left(\frac{3}{5}\right)^k\right)$

$\leq \frac{5}{2}n \in O(n)$

Question 4 . . . . 4 points

Consider the following algorithm.

Algorithm 1 Unknown($A[0..n-1]$) ▷ Input: An array $A[0..n-1]$ of n real numbers

```
minval ← A[0]
maxval ← A[0]
for i ← 1, n − 1 do
    if A[i] < minval then
        minval ← A[i]
    end if
    if A[i] > maxval then
        maxval ← A[i]
    end if
end for
return maxval − minval
```

(a) [1 point] What does this algorithm compute?

It gives inital value A[0] to minval & maxval then it goes through the entire array. If a ~~[illegible]~~ number lower than minval is found, That number (A[i]) is the new minval. Same for maxval except if a larger number (A[i]) is found larger than maxval. The number (A[i]) is the new maxval. At last, it returns the difference between maxval & minval.

(b) [1 point] What is its basic operation?

Comparison (<, >)

(c) [2 points] Give the best-case and worst-case time complexities of this algorithm in asymptotic notation.

Best case = O(n)

worst case = O(n)

It will loop the entire array either way. n terms so O(n).

Question 5 . . . . 6 points

Consider the following recursive algorithm for computing the sum of the first n squares of integers: $S(n) = 1^2 + 2^2 + ... + n^2$.

Algorithm 2 $S(n)$ ▷ Input: A positive integer n ▷ Output: the sum of the first n squares

```
if n = 1 then
    return 1
else
    return S(n − 1) + n ∗ n
end if
```

(a) [½ point] What is the algorithm's basic operation?

Addition ~~and~~ or Multiplication (+, *)

(b) [2 points] Set up and solve a recurrence relation for the number of times the algorithm's basic operation is executed.

$S(n) = S(n-1) + 1$
$= S(n-2) + 1 + 1$
$= S(n-3) + 1 + 1 + 1$

$S(n-k) + k$
stop when = 1
$n - k = 1$
$n = k + 1$

$\Rightarrow S(1) + k = 1 + k = n \in O(n)$

(c) [2 points] Give a pseudocode of a non-recursive version of this algorithm.

```
S(n){
    int i ← 1                    1
    int sum ← 0                  1
    for i ≤ n do {               n+1
        sum ← sum + i*i          n
    }
    return sum                   1
}
```

(d) [½ point] What is the worst-case time complexity of the algorithm in (c)?

O(n). It goes through the lop n times.

$2n + 4 \in O(n)$

(e) [1 point] What is the suitable algorithm for computing $S(n)$ (recursive or non-recursive)? Justify your answer.

Both are O(n) same time-complexity. The non-recursive uses less memory, so it is the better choice.

Question 6 .... 4 points

(a) [3 points] Describe an algorithm that takes a list of n positive integers and finds the location of the last even integer in the list. It returns 0 if there are no even integers in the list.

```
find even (A[0...n-1]) {
    i ← n-1
    while (i ≥ 0) do {
        if (A[i] % 2 = 0)
            return i
        i--
    }
    return 0
}
```

(b) [1 point] Give the best-case and worst-case time complexities of your algorithm in asymptotic notation. Show all details.

~~Best case = worst case = O(n)~~
~~The algorithm will go through n terms in a for loop regardless if there's any even integers.~~

Best case = O(1) if A[n-1] is even.
Worst case = O(n) if there is no even integer in list.